部署报告

16340304　　郑国林

1.选题

我的题目确认为利用区块链实现一个轻量型的微博系统。

2.合约编写

分成了两个合约，一个合约是总体的合约，在一开始先布置，然后用户在注册的时候再分别为每个用户单独部署一个合约。

MyBlog 合约中存储了所有账户的信息，对应地址到名字，对应地址到注册时部署的 Account 合约。里面实现的接口 register 是通过名字注册，会检测调用该函数的地址是否已经注册过账号，或者是该名字是否已经被注册，如果都没有，就添加该名字和地址到对应的 mapping 中。 setDeployedAdd 是将注册时在 js 中部署的 account 地址存到对应名字的 mapping 中， getDeployedAdd 是通过名字获取该名字对应的 account 合约地址， getAccountNum 是获取全部账号数量的接口， getAddByName 是通过名字来获取注册的用户地址， getNameByAdd 是通过注册用户地址来获取注册的用户名。

Account 合约是管理每个账户发布微博的合约。里面为每个微博定义了一个结构体，包含文字信息，发布时间信息，点赞数和不点赞数，还有是否私有的微博标志。要初始化合约拥有者为部署合约的地址，方便后面的接口不被其他人调用。 并且定义了一个 blogId 到 blog 的 mapping。publish 是发布微博的接口，只有 owener 能够调用，要求发布的文字不能够超过 100 个字符，并将 id 唯一赋值给这个 blog，并在其中初始化 blog 的点赞数，是否为私有等数据。like 接口是点赞别人微博的接口，通过别的用户的名字可以获取到它的 account 合约的地址，然后调用其 like 接口对某个 id 的微博点赞。dislike 和 like 接口基本相同。getBlog 是获取对应 id 的 blog 的详细信息，像文字，点赞数等，但是如果为私有的只有 owener 可以获取到。getBlogNum 是获取 blog 数量的接口。我的想法是先获取 blog 数量，然后遍历就可以显示所有的 blog。至于删除的部分还没有写好，大概的想法是在 Blog 结构体中增加一个标志位来标志是否是删除的，不然在遍历从 0 到 blogNum 显示微博的时候会出错。

两个合约的代码如下：

```
pragma solidity ^0.4.17;

contract MyBlog {

  mapping (address => string) nickNames;
  mapping (string => address) registerNames;
  mapping (string => address) deployedContracts;
  uint totalAccounts;

  address admin;
  modifier onlyRegistryAdmin
  {
    require(msg.sender == admin);
    _;
  }

  function MyBlog() public
  {
    admin = msg.sender;
    totalAccounts = 0;
  }

  function register(string name) public
  {
    address add = msg.sender;
```

```
    //require(registerNames[name] == address(0));
    //require(bytes(nickNames[add]).length == 0);

    nickNames[add] = name;
    registerNames[name] = add;
    totalAccounts++;
  }

  function setDeployedAdd(string name, address add) public
  {
    require (msg.sender == registerNames[name]);
    deployedContracts[name] = add;
  }

  //can used by login or someother want to get by others name
  function getDeployedAdd(string name) public constant returns (address contractAdd)
  {
    return deployedContracts[name];
  }

  function getAccountNum() public constant returns (uint _accountNum)
  {
    return totalAccounts;
  }

  function getAddByName(string name) public constant returns (address add)
  {
    return registerNames[name];
  }

  function getNameByAdd(address add) public constant returns (string name)
  {
    return nickNames[add];
  }

}



pragma solidity ^0.4.17;

contract Account {
  struct Blog
  {
    string context;
    uint publishTime;
    uint likeNum;
    uint dislikeNum;
    bool _private;
  }

  mapping (uint => Blog) blogs;
```

```
  uint blogNum;
  address owner;
  modifier onlyOwner
  {
    require(msg.sender == owner);
    _;
  }

  function Account() public
  {
    blogNum = 0;
    owner = msg.sender;
  }

  function publish (string context, bool _private) public onlyOwner
  {
    require(bytes(context).length <= 100);

    blogs[blogNum].publishTime = now;
    blogs[blogNum].context = context;
    blogs[blogNum].likeNum = 0;
    blogs[blogNum].dislikeNum = 0;
    //default is public
    blogs[blogNum]._private = _private;
    blogNum++;
  }

  function like (uint blogId) public
  {
    require(blogId < blogNum);
    blogs[blogId].likeNum++;
  }

  function dislike (uint blogId) public
  {
    require(blogId < blogNum);
    blogs[blogId].dislikeNum++;
  }

  function getBlog (uint blogId) public constant returns (string context, uint publishTime, uint
likeNum, uint dislikeNum)
  {
    require(blogId < blogNum);

    if(blogs[blogId]._private)
      require(msg.sender == owner);

    context = blogs[blogId].context;
    publishTime = blogs[blogId].publishTime;
    likeNum = blogs[blogId].likeNum;
    dislikeNum = blogs[blogId].dislikeNum;
  }
```

```
  function getBlogNum() public constant returns (uint _blogNum)
  {
    return blogNum;
  }
}
```

3.合约部署情况

我用了 truffle 和 ganache 来帮助开发，用 ganache 来开一个用于测试的链，truffle 可以帮助我们快速进行合约的部署和测试，到测试到差不多才部署到私链上。部署到私链的过程和部署到 ganache 是差不多的，truffle 的 migrate 命令会根据 truffle.js 中的端口和 netid 来进行部署。

首先打开 ganache 会开一个测试链（或者打开 geth 的私有链）

```
artemis@artemis-OMEN-by-HP-Laptop:~/test$ ganache-cli >> trace.log
```

会自动创建了 10 个账户给我们进行测试，每个账户中有 100 个币

```
Available Accounts
==================
(0) 0x9a260142212507f87fc910bef9c692984d086752 (~100 ETH)
(1) 0xd6ba8baa6b73f005909c1c2a1db74152f5055771 (~100 ETH)
(2) 0x67c0a708cdb50d1d8a4f90fc8289067d0c5ab11b (~100 ETH)
(3) 0x67af9937634b90c29f5ab7f0a60dab4ae6615da1 (~100 ETH)
(4) 0x34e9e77c76ac721b2a05ab974e940c78802e7764 (~100 ETH)
(5) 0x615462722da9c207fcea72a8b1b03926697998b2 (~100 ETH)
(6) 0x73acb6df1a5ce072b5795565a52e8e6d6742ea23 (~100 ETH)
(7) 0x4484479e17434d3f99cc361f2b2e740a2ed56921 (~100 ETH)
(8) 0xf6d8f40ea103edb67abc1c00c7938602456fff7f (~100 ETH)
(9) 0xfaf74ae50715de5b1a9119c6d080cc623b4760fa (~100 ETH)

Private Keys
==================
(0) 0x1e81848a9e64f9771bc9f7c9a23b9db4fef4d87467fc3e7bd81297f0898fd7d3
(1) 0x1fa15aa40e6c6049cd67fe5f0840819007475abfdac65f39f23eef0468e24222
(2) 0xdb82817e63d04e7c3456677403ed262f1b4a43b3367e01469e4a9393f02d0590
(3) 0x2f315651bac64b5ef7107b579bb9946a4e575cc0cb3376aac184486171fcb2c2
(4) 0x8f66c3d622658be25fbf07a24c8fa89a637d3d0aad8ac86ad87a0bfe27bdfb8b
(5) 0x2d3c3eeeb8af61bb6b63eb144bddb06942a5a32e01071924bc58c41ed1a20897
(6) 0xb389efeacc581f64a7bf69d7668419015df2015f237e08562439ec7bf49d32e0
(7) 0xe447f6d258e5f5df78e5ae2d5cbda1faf00a84df2762b39d597fe87ab1479c1d
(8) 0xf9f159882d75f7ae9a09008d21a458bef0a2cf04b8e12dcebcedf7ca87cc04be
(9) 0xecc792c0ceaa2d536cfd433761280f960ad39b567dd74d85afcc9061775fed52
```

然后用 truffle 先编译 truffle compile

```
artemis@artemis-OMEN-by-HP-Laptop:~/test$ truffle compile
artemis@artemis-OMEN-by-HP-Laptop:~/test$
```

就会把编译后的 json 文件写到 contracts 目录下，再通过 truffle migrate 来把 MyBlog 合约部署到测试的链上，Account 合约的部署我写在了 js 上，是要用户点击的时候再部署的。

```
artemis@artemis-OMEN-by-HP-Laptop:~/test$ truffle migrate
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x1a1637b834538be8cbdac5df0bc9b8257ae73a4a6f3a28aa99df68930621a1ab
  Migrations: 0x140526a5bd2dfd5328cc3129c0137d5ecd49d1b0
Saving successful migration to network...
  ... 0x51a1128ed7a6455a04d0513ba004849cf3ee15c41cb48c59b0c70cbd13f9ec4a
Saving artifacts...
Running migration: 2_myBlog_migration.js
  Deploying MyBlog...
  ... 0x37cf755e0c872e8b07ec2813602d41cd0a10a1d0c249447054eef6e61fbbd9b1
  MyBlog: 0x862050ea8f0efd855e7b0e3c77050ef33f1ddb9b
Saving successful migration to network...
  ... 0xc05e200eb353e31e5db5660e8abbd454764afcb8f3003ee5ac053defd1f9171f
Saving artifacts...
```

用 truffle migrate 部署要写一段代码，如下

```
var MyBlog = artifacts.require("MyBlog");

module.exports = function(deployer) {
  deployer.deploy(MyBlog);
};
```

可以从 log 文件中看到部署成功

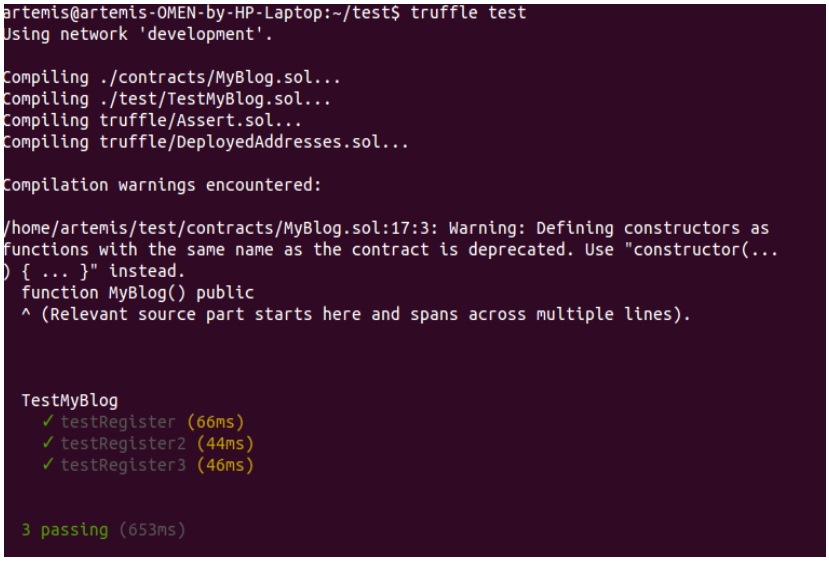
```
eth_sendTransaction

  Transaction: 0x1a1637b834538be8cbdac5df0bc9b8257ae73a4a6f3a28aa99df68930621a1ab
  Contract created: 0x140526a5bd2dfd5328cc3129c0137d5ecd49d1b0
  Gas usage: 277462
  Block Number: 1
  Block Time: Mon Nov 26 2018 22:51:32 GMT+0800 (CST)

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_sendTransaction

  Transaction: 0x51a1128ed7a6455a04d0513ba004849cf3ee15c41cb48c59b0c70cbd13f9ec4a
  Gas usage: 42008
  Block Number: 2
  Block Time: Mon Nov 26 2018 22:51:32 GMT+0800 (CST)

eth_getTransactionReceipt
eth_accounts
net_version
net_version
eth_sendTransaction

  Transaction: 0x37cf755e0c872e8b07ec2813602d41cd0a10a1d0c249447054eef6e61fbbd9b1
  Contract created: 0x862050ea8f0efd855e7b0e3c77050ef33f1ddb9b
  Gas usage: 703935
  Block Number: 3
  Block Time: Mon Nov 26 2018 22:51:32 GMT+0800 (CST)

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_sendTransaction

  Transaction: 0xc05e200eb353e31e5db5660e8abbd454764afcb8f3003ee5ac053defd1f9171f
  Gas usage: 27008
  Block Number: 4
  Block Time: Mon Nov 26 2018 22:51:32 GMT+0800 (CST)
```

接着可以用命令 truffle test 进行测试，要先写测试的代码放在 test 目录下，我是测试了注册是否能成功，是否能通过接口获取到注册的账户地址和名字。测试的代码和结果如下：

```
pragma solidity ^0.4.17;

import "truffle/Assert.sol";   // 引入的断言
import "truffle/DeployedAddresses.sol";  // 用来获取被测试合约的地址
import "../contracts/MyBlog.sol";      // 被测试合约

contract TestMyBlog {
  MyBlog myBlog = MyBlog(DeployedAddresses.MyBlog());


  function testRegister() public {

    myBlog.register("zgl");
```

```
    uint accountNum = myBlog.getAccountNum();
    uint expected = 1;
    Assert.equal(accountNum, expected, "the first register should be recorded.");
  }

  function testRegister2() public {

    address add = myBlog.getAddByName("zgl");
    address expected = this;
    Assert.equal(add, expected, "add should record");
  }

  function testRegister3() public {

    Assert.equal(myBlog.getNameByAdd(this), "zgl", "name should record");
  }
}
```

测试结果

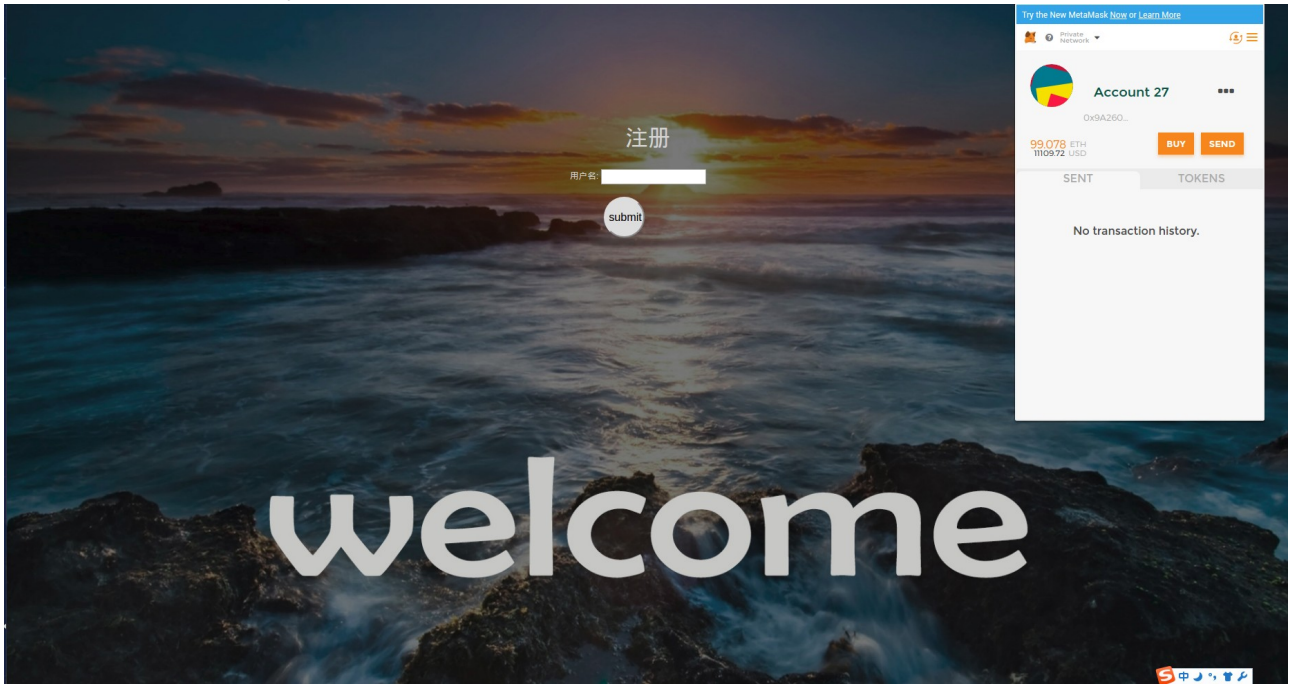


```
artemis@artemis-OMEN-by-HP-Laptop:~/test$ truffle test
Using network 'development'.

Compiling ./contracts/MyBlog.sol...
Compiling ./test/TestMyBlog.sol...
Compiling truffle/Assert.sol...
Compiling truffle/DeployedAddresses.sol...

Compilation warnings encountered:

/home/artemis/test/contracts/MyBlog.sol:17:3: Warning: Defining constructors as
functions with the same name as the contract is deprecated. Use "constructor(...
) { ... }" instead.
  function MyBlog() public
  ^ (Relevant source part starts here and spans across multiple lines).



  TestMyBlog
    ✓ testRegister (66ms)
    ✓ testRegister2 (44ms)
    ✓ testRegister3 (46ms)


  3 passing (653ms)
```

在 log 文件中也可以开到测试时的交易记录

```
eth_sendTransaction

  Transaction: 0x4ace633584894a34a92bf33d98e8a487f495bc7eeb16371eef2785632c4221af
  Contract created: 0x55872d5f0832e276e2364bc8b78a22ad673fb1af
  Gas usage: 650149
  Block Number: 11
  Block Time: Mon Nov 26 2018 22:59:39 GMT+0800 (CST)

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_blockNumber
eth_sendTransaction

  Transaction: 0x615753636a8193fd393016f9869e6f6a7bbb61eef24071be8c5948f67920a984
  Gas usage: 92662
  Block Number: 12
  Block Time: Mon Nov 26 2018 22:59:39 GMT+0800 (CST)

eth_getTransactionReceipt
eth_blockNumber
eth_sendTransaction

  Transaction: 0xf3b08d172992a61eaf30e51b5c9dfd56a7dcd23999720a13231dc6b6f5ea9c4d
  Gas usage: 29686
  Block Number: 13
  Block Time: Mon Nov 26 2018 22:59:39 GMT+0800 (CST)

eth_getTransactionReceipt
eth_blockNumber
eth_sendTransaction

  Transaction: 0xa85b0d4ffa8b2f9a1075f1e5d64bf67d25d283d1fb463c8a3d2b1c1269e4798c
  Gas usage: 32039
  Block Number: 14
  Block Time: Mon Nov 26 2018 22:59:40 GMT+0800 (CST)
```

接着我使用 npm 装了一个 lite-server 来管理网页的静态文件，写了个 js 和简单的网页来测试一下能否在 js 中部署合约，首先还要在浏览器中安装 metamask 插件。通过 metamask 可以导入我们的账户，然后可以很方便的确认交易。



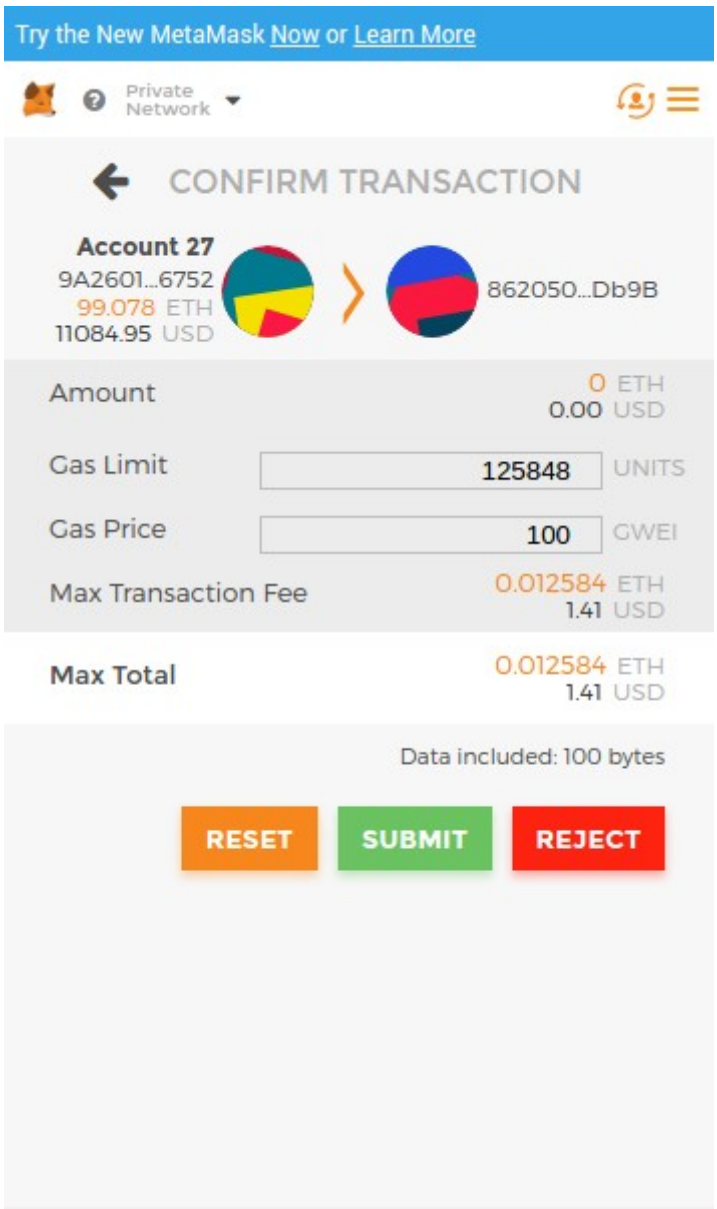
然后在 js 中我们要先设置 web3Provider，然后获取我们在 truffle 中已经部署的合约（有函数接口可以用），然后绑定一些事件处理函数，并且在点击注册是再部署一个 Account 合约（这部分是比较重要的），这时候就要用到 web3 和已经编译好的 json 文件中的 abi 和 bytecode 了，具体的代码如下：

```
App = {
	web3Provider: null,
	contracts: {},

	init: async function(){

		return await App.initWeb3();
	},

	initWeb3: async function(){
		if (window.ethereum) {
			App.web3Provider = window.ethereum;
			try{
				await window.ethereum.enable();
			}catch(error){
				console.error("User denied account access");
			}
		}
		else if (window.web3) {
			App.web3Provider = window.web3.currentProvider;
		}
		else{
			App.web3Provider = new
Web3.providers.HttpProvider('http://localhost:8545');
		}
		web3 = new Web3(App.web3Provider);

		return App.initContract();
	},

	initContract: function(){
		$.getJSON('MyBlog.json', function(data){
			var MyBlogArtifact = data;
			App.contracts.MyBlog = TruffleContract(MyBlogArtifact);
			App.contracts.MyBlog.setProvider(App.web3Provider);
			return;
		});
		return App.bindEvents();
	},

	bindEvents: function(){
		//to be continue
		$(document).on('click', '#submit', App.submitClick);
	},

	submitClick: function(event){
		event.preventDefault();
```

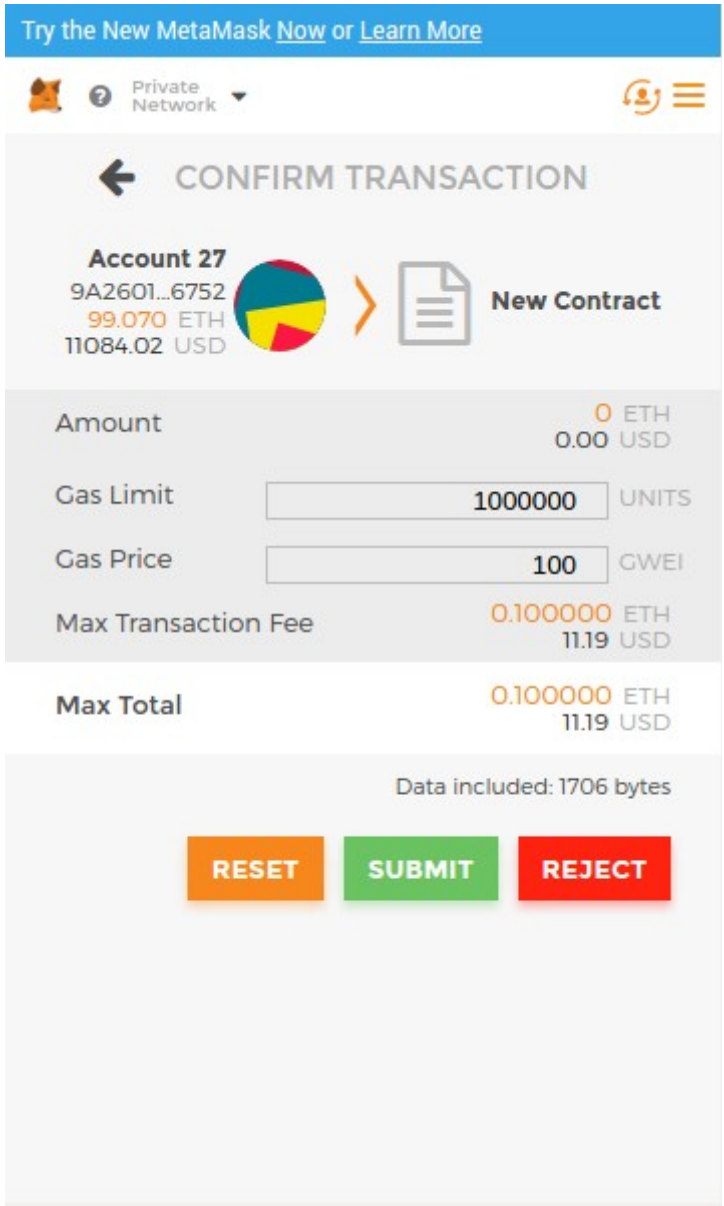
```
        var name = $('#nameinput').val();
        console.log(name);

        var MyBlogInstance;

        web3.eth.getAccounts(function(error, accounts){
                if(error){
                        console.log(error);
                }

                var account = accounts[0];

                App.contracts.MyBlog.deployed().then(function(instance){
                        MyBlogInstance = instance;
                        return MyBlogInstance.register(name, {from: account});
                }).then(function(result){
                        return App.deployAccount(name);
                }).catch(function(err){
                        console.log(err.message);
                });
        });

},

deployAccount: function(name){
        var MyBlogInstance;
        var Account;
        var AccountInstance;
        var add;

        web3.eth.getAccounts(function(error, accounts){
                if(error){
                        console.log(error);
                }

                var account = accounts[0];
                var AccountABI;
                var bytecode;
                $.getJSON('Account.json', function(data){
                        AccountABI = data.abi;
                        bytecode = data.bytecode;
                        console.log(AccountABI);
                        console.log(bytecode);
                        var AccountContract = web3.eth.contract(AccountABI);
                        console.log(AccountContract);
                        var ContractInstance = AccountContract.new({
                           data: bytecode,
                           from: account,
                           gas: 1000000
                        },function (e, contract){
                           console.log(e, contract);
                           if (typeof contract.address !== 'undefined') {
```

```
                console.log('Contract mined! address: ' + contract.address + '
transactionHash: ' + contract.transactionHash);

                App.contracts.MyBlog.deployed().then(function(instance){
                        MyBlogInstance = instance;
                        return MyBlogInstance.setDeployedAdd(name,
contract.address, {from: account});
                }).then(function(result){


                        window.location.href =
'tmnt.html?'+"name="+encodeURI(name);
                        return;
                }).catch(function(err){
                        console.log(err.message);
                });
            }
        }
        );
        return;
    });
  });
 }


};

$(function() {
  $(window).load(function() {
    App.init();
  });
});
```

然后在网页上进行注册
会弹出确认交易，第一个交易是调用 MyBlog 的 Register 接口，用来保存用户名和注册的用户地址



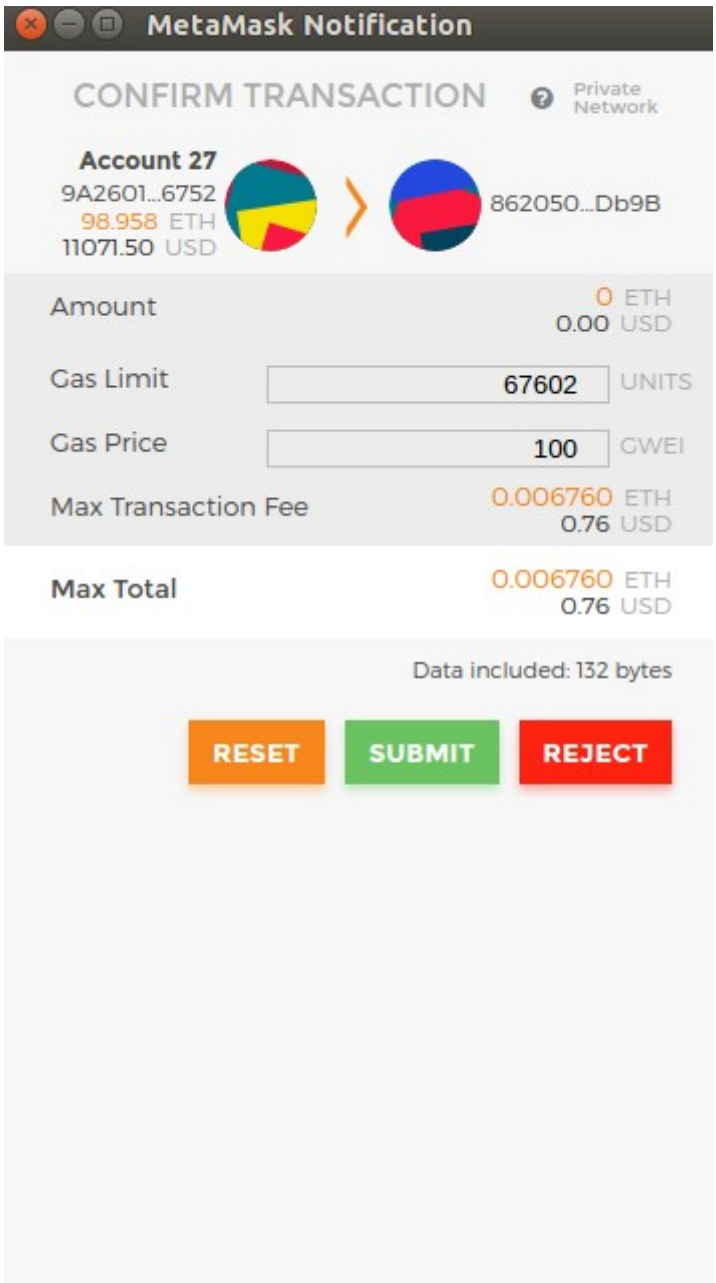
第二个交易是部署 Account 合约



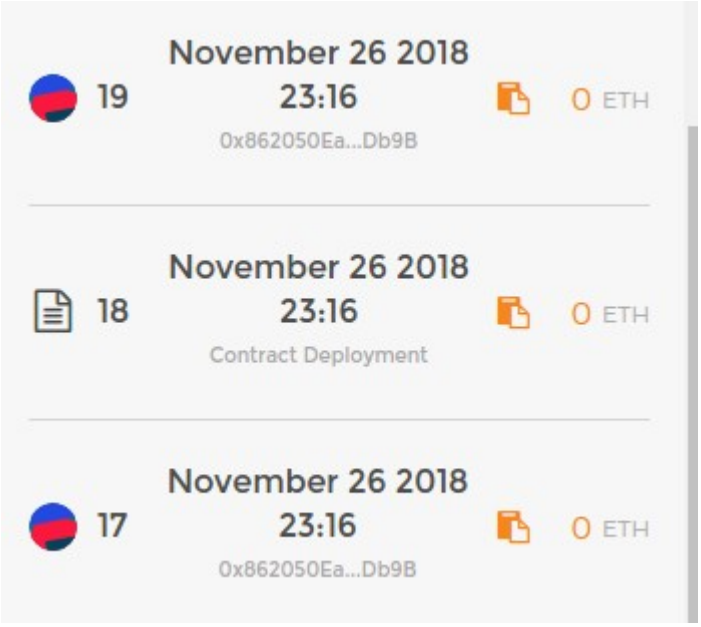
通过 consonle 可以看到部署成功的合约地址

```
Contract mined! address: 0x5fa03b37c88716f7792751a0cfa694c7281fcae8 transactionHash:
0x79c3ff307cd38d6e9f81b0598f564411913e7752d9117656688bf6d9499c943c
```

第三个交易是调用 MyBlog 的 setDeployedAdd 接口来设置该用户部署的 accout 合约地址（下次登录直接获取就可以进行操作了）



交易是否成功也可以看到



由于后面的页面没写好，对 Accout 的测试就通过 remix 的 web3 部署到测试链上进行测试了一下

对 pulish 接口的测试，发布了一个微博

[block:22 txIndex:0] **from:**0x9a2...86752 **to:**Account.publish(string,bool) 0xc57...7705c **value:**0 wei **data:**0x21b...00000 **logs:**0 **hash:**0x581...c87bd

status	0x1 Transaction mined and execution succeed
transaction hash	0x5815147039ad9b4d1381c0a225642ba21eaf15135e4d0d38c54542ded9dc87bd
from	0x9a260142212507f87fc910bef9c692984d086752
to	Account.publish(string,bool) 0xc570acfb9d996001a74dfe0965e5c517e237705c
gas	100262 gas
transaction cost	100262 gas
hash	0x5815147039ad9b4d1381c0a225642ba21eaf15135e4d0d38c54542ded9dc87bd
input	0x21b...00000
decoded input	{ "string context": "abc", "bool _private": false }
decoded output	-
logs	[]
value	0 wei

对 getBlogNum 测试,结果为 1

call to Account.getBlogNum

[call] **from:**0x9a260142212507f87fc910bef9c692984d086752 **to:**Account.getBlogNum() **data:**0xf9b...052cd Debug

transaction hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xf9b052cd
from	0x9a260142212507f87fc910bef9c692984d086752
to	Account.getBlogNum() 0xc570acfb9d996001a74dfe0965e5c517e237705c
hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xf9b052cd
input	0xf9b...052cd
decoded input	{}
decoded output	{ "0": "uint256: _blogNum 1" }
logs	[]

对 getBlog 测试，注意 blogId 从 0 开始，获取正常

[call] **from:**0x9a260142212507f87fc910bef9c692984d086752 **to:**Account.getBlog(uint256) **data:**0xd48...00000 Debug

transaction hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xd4868baa0000000000000000000000000000000000000000000000000000000000000000
from	0x9a260142212507f87fc910bef9c692984d086752
to	Account.getBlog(uint256) 0xc570acfb9d996001a74dfe0965e5c517e237705c
hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xd4868baa0000000000000000000000000000000000000000000000000000000000000000
input	0xd48...00000
decoded input	{ "uint256 blogId": "0" }
decoded output	{ "0": "string: context abc", "1": "uint256: publishTime 1543246330", "2": "uint256: likeNum 0", "3": "uint256: dislikeNum 0" }
logs	[]

对 like 进行测试，再获取该 blog，likeNum 变成 1，成功进行点赞。

[call] **from:**0x9a260142212507f87fc910bef9c692984d086752 **to:**Account.getBlog(uint256) **data:**0xd48...00000 Debug

transaction hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xd4868baa0000000000000000000000000000000000000000000000000000000000000000
from	0x9a260142212507f87fc910bef9c692984d086752
to	Account.getBlog(uint256) 0xc570acfb9d996001a74dfe0965e5c517e237705c
hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xd4868baa0000000000000000000000000000000000000000000000000000000000000000
input	0xd48...00000
decoded input	{ "uint256 blogId": "0" }
decoded output	{ "0": "string: context abc", "1": "uint256: publishTime 1543246330", "2": "uint256: likeNum 0", "3": "uint256: dislikeNum 0" }
logs	[]

[call] **from:**0x9a260142212507f87fc910bef9c692984d086752 **to:**Account.getBlog(uint256) **data:**0xd48...00000

transaction hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xd4868baa0000000000000000000000000000000000000000000000000000000000000000
from	0x9a260142212507f87fc910bef9c692984d086752
to	Account.getBlog(uint256) 0xc570acfb9d996001a74dfe0965e5c517e237705c
hash	call0x9a260142212507f87fc910bef9c692984d0867520xc570acfb9d996001a74dfe0965e5c517e237705c0xd4868baa0000000000000000000000000000000000000000000000000000000000000000
input	0xd48...00000
decoded input	{ "uint256 blogId": "0" }
decoded output	{ "0": "string: context abc", "1": "uint256: publishTime 1543246330", "2": "uint256: likeNum 1", "3": "uint256: dislikeNum 0" }
logs	[]